Computational Steering : a Case Study

Robert van Liere
Department of Interactive Systems
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
e-mail: robertl@cwi.nl

We discuss the application of interactive visualization techniques to a class of problems where a user interacts with an underlying mathematical model and steers its simulation in the hope of "gaining new insight". The primary interface for this interaction are visual images and the subsequent input based upon the user's perception and interpretation of these images. The emphasis is on applications which involve physical models in higher-dimensional space; in particular, we focus on the simulation, visualization and interaction of chemical reactions.

In this paper we present several aspects of our ongoing work in interactive visualization. After introducing simple underlying concepts, we discuss some issues that we are studying.

1. Introduction

Before we delve into a discussion about scientific visualization and how it is done, it is useful to understand the reason why users are finding interactive visualization tools indispensable for their daily work. From the point of view of the end user, there are three fundamental problems currently plaguing large scale numerical simulations:

- large volumes of output data can be produced from a relatively small amount of input data;
- data can be produced faster than it can be stored onto a mechanical device, such as disk;
- data can be produced and stored faster than a user can comprehend its significance.

The overwhelming amount of data generated by these large scale numerical simulations makes it impossible for users to quantitatively examine more than a tiny fraction of a given solution. Scientific visualization is an emerging interdisciplinary field that is responding to the need for insight into data representation through creation of visualization tools. The purpose of scientific visualization is thus to enhance existing scientific and numerical methods by increasing a scientist's ability to see data and comprehend the results of computations. The conveyed information undergoes a qualitative change because it brings the eye-brain system, with its great pattern-recognition capabilities, into play in a way that is impossible with purely numerical data.

As a brief introduction to the role that visualization plays within simulations and experiments, we introduce a reference pipeline which acts as a framework to explain the visualization process. The primary interface for interaction between user and simulation are the visual images as presented through the rendering process and the subsequent input based upon the user's perception and interpretation of these images.

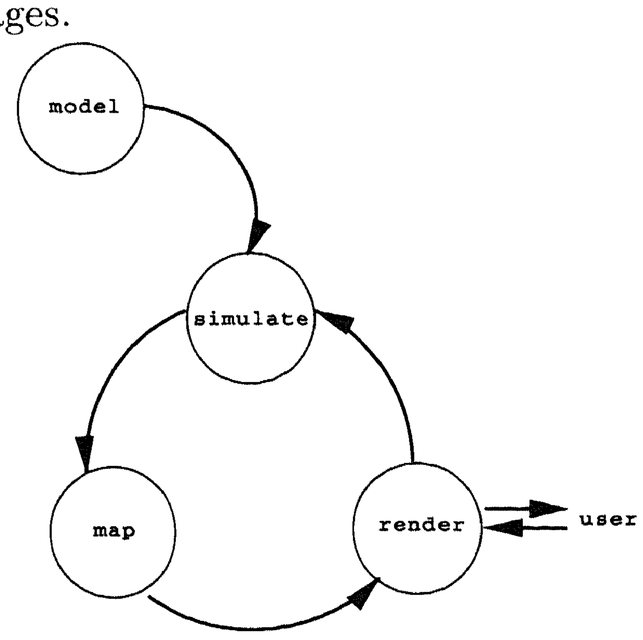


FIGURE 1. The visualization pipeline.

The visualization pipeline has four major phases:

- The *modeling* phase, which denotes the process of finding a mathematical model of some physical phenomenon occurring in the real world. In general, the resulting model is in the form of a set of partial differential equations and as such is not capable of being solved numerically. In some limited cases the model can have an exact analytic solutions, although this is rare.

- The *simulation* phase denotes the simulation of the mathematical model after it has been approximated by various discretization techniques. The discretization of space represents a sampling of the real world at a small number of selected points. Thus, it is the interface between that which is not modeled and that which must be handled with care. To satisfy these constraints, boundary conditions must be established to handle spatial approximation of the rest of the world, and initial conditions to approximate the temporal conditions leading up to the time the simulation starts.

- The *data mapping* phase is responsible for preparing the raw simulated data to geometric primitives which can effectively convey the information content of data visually. Each set of data can be transformed onto a broad array of geometric primitives. For example, a three-dimensional scalar field can be visualized using everything from zero-dimensional point primitives to three-dimensional volumetric primitives. Each representation form reveals different aspects of the information contained within the data set. Every geometric primitive type has a concomitant set of surface or volumetric properties assigned to it which give the primitive distinguished visual characteristics.

  There is a trade-off between an informative and an economical representation when choosing a particular geometric primitive to represent a data set. For three-dimensional data sets it is possible to generate up to an order of magnitude more data in the form of geometric primitives than the raw data they represent. The generation of a large number of primitives can also be very compute intensive, as computationally demanding as the simulation itself.

  The mapping phase is the key to the visualization pipeline, and the natural juncture of scientific and graphical data.

- The *presentation* phase which produces the actual image using various rendering operations like projection, shading, texture mapping and hidden surface removal. The rendering phase is the easiest of the entire visualization process to define. In general, the rendering techniques used in the computational sciences are a small subset of those used in the entertainment and television computer animation industry, since the scientists' goals are not visual realism but scientific insight. This property permits several of the commonly used algorithms to be implemented in hardware, such as z-buffering of points, lines and polygons, sphere rendering, and volumetric rendering. While a hardware implementation is more restrictive and currently of lower quality than that achievable in software, it is much faster. In the computational sciences, interactive speed is more important than elaborate illumination models.

Due to the exploratory character of the visualization process, the pipeline is not organized as a set of linear phases but as a cycle in which a user can interact with previous phases. Computational steering is a form of interaction which provides a user, through the manipulation of a visual representation of the ongoing calculation, extensive control over a numerical calculation. As an illustrative example consider the situation in which steering gives a user the opportunity to interrupt a simulation of a physical phenomenon, query or mutate some parameters and then continue the simulation. Examples of the parameters that a user may want to manipulate interactively are spatial grid definitions, manipulation of temporal aspects of simulation, and the specifications of other relevant initial conditions. In this situation, visual representations are responsible to convey

the behavior of the simulated mathematical model.

Providing users with effective steering techniques will dramatically shorten the modeling/simulation/analysis cycle because these techniques allow users to "directly manipulate" the underlying simulation. Furthermore, computational steering provides the basis for visual modeling and analysis environments, in which users (a) iteratively define and refine a model of some observed physical phenomenon and (b) incrementally analyze the results of a simulation.

Readers are encouraged to read [8] and [5] for a more complete introduction to visualization techniques and interactive visualization.

2. Parameter Estimation in Non-linear Differential Equations

We study computational steering within the application framework of kinetic reactions of chemical processes. Complex chemical reactions are modeled as a parameterized set of differential equations which, given a number of conditions, can be solved with well known numerical techniques. However, in order to get a quantitative and mathematical understanding of the model being investigated, a chemist is not only interested in the most optimal values of the parameters, but also in issues such as the reliability and stability of the model.

2.1. Problem formulation

The mathematical model is formulated as a parameter estimation problem:

$$\begin{cases} \boldsymbol{y}'(t,\boldsymbol{p}) &= \boldsymbol{G}\ (t,\boldsymbol{y},\boldsymbol{p}) \\ \boldsymbol{y}(t_0,\boldsymbol{p}) &= \boldsymbol{y}_0(\boldsymbol{p}) \end{cases} \tag{1}$$

where the variable vector is denoted as: $\boldsymbol{y} : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, and the system of equations as : $\boldsymbol{G} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, in which $t \in \mathbb{R}$ and $\boldsymbol{p} \in \mathbb{R}^m$ represents the vector of parameters.

In this formulation the parameter vector $\boldsymbol{p}$ cannot be determined. However, assume that $\{(t_i, y_i^{c_i})\}_{i=1}^k$, are k observations of the chemical reaction, at time t_i for the c_i-th component of the variable vector, $\boldsymbol{y}$. The parameter vector, $\boldsymbol{p}$, is a solution of the *inverse problem*, if the following equation holds :

$$y^{c_i}(t_i,\boldsymbol{p}) - y_i^{c_i} = 0, \quad i = 1,\ldots,k. \tag{2}$$

In general this equation cannot be met; rather, curve fitting techniques are applied which minimize the difference. The inverse problem can thus be stated as: determine $\boldsymbol{p}$, such that

$$\left|\hat{\boldsymbol{Y}}(\boldsymbol{p}) - \boldsymbol{Y}\right| \text{is minimal ,} \tag{3}$$

in which:

$$\hat{\boldsymbol{Y}}(\boldsymbol{p}) = \left(y^{c_1}(t_1,\boldsymbol{p}),\ldots,y^{c_k}(t_k,\boldsymbol{p})\right)^{\mathrm{T}},$$

and

$$\boldsymbol{Y} = (y_1^{c_1}, \ldots, y_k^{c_k})^{\mathrm{T}} .$$

A particular minimization criterion, and the one chosen in the current implementation, is the sum of the squares of the discrepancies between the calculated values and the observed values; i.e.

$$S(\boldsymbol{p}) = \left\| \hat{\boldsymbol{Y}}(\boldsymbol{p}) - \boldsymbol{Y} \right\|_2^2 . \tag{4}$$

Standard numerical techniques can be used to find vectors $\boldsymbol{p}$ that minimize $S(\boldsymbol{p})$. Details of these solution methods can be found in [1].

2.2. Discussion

The inverse problem is a very attractive problem to use as an application framework to study computational steering. Even if we assume that there is a unique minimum and that the function $S(\boldsymbol{p})$ is the best one to minimize, there still remain many questions the chemist would like to pose. For example :

- what influence do perturbed observations, $y_i^{c_i}$, have on the minimization function, $S(\boldsymbol{p})$?
- what is the confidence interval of the found parameter vector, $\boldsymbol{p}$?
- are there dependencies between individual parameters $p_i \in \boldsymbol{p}$ and, if so, what are these ?

Although visualization is a powerful tool to augment the numerical techniques, visual representations by themselves are not enough. Even with the best display techniques, it is impossible to simultaneously display all of the quantitative information in a large data set in an understandable form. When more precise information on some aspect of the data is required, it must be specifyed through user interaction. This interaction can include simple refinements of the color coding of contour levels or complex manipulation of virtual probes that simulate the physical behavior or measurements. We conjecture that visualization techniques must be embedded within the computation so that they may be used when developing and debugging the underlying model. This calls for a very intergrated and interactive environment in which users are able to attach probes and monitor data, in much the same way as oscilloscopes and other kinds of measuring instruments are used in experimental sciences. Eventually, these interactive visualization techniques will allow users, through visual specification and interrogation, to directly manipulate the simulation process itself. Examples of such techniques are:

- visual specifications of dependencies between parameters and immediately see the effect that this has on the simulation.

- to visually apply reconciliation techniques on the collected data and immediately see the effect.

3. Visualization Techniques

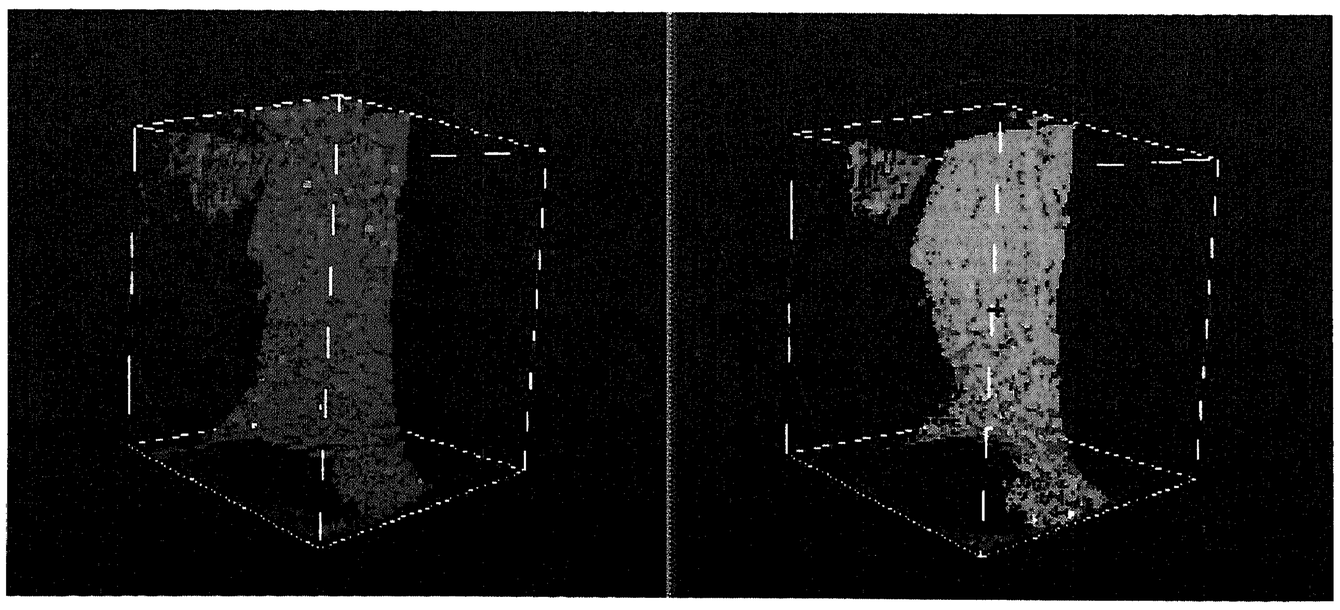
We will give two characteristic examples of visualization techniques: one which discusses how $S(\boldsymbol{p})$ can be visualized and the second on how $S(\boldsymbol{p})$ can interactively be constructed. These examples are meant only to illustrate some issues on how visualization can be used to steer a complex numerical computation.

3.1. Parameter space representation

We use isosurfaces to visualize the minimalization function, $S(\boldsymbol{p})$, with $\boldsymbol{p} = (p_1, p_2, \ldots, p_m)^{\mathrm{T}}$ and $p_i \in \mathbb{R}$. Isosurfaces provide important cues for visualizing and interpreting 3-D scalar fields. For any scalar field, $f(x, y, z)$, an isosurface is defined as the set of points satisfying $f(x, y, z) = c$ for some constant c, known as the isovalue. The well known Marching Cube algorithms can be used to compute an isosurface as a set of polygons, [4].

Assume $\hat{\boldsymbol{p}} \in \mathbb{R}^3 \subset \mathbb{R}^m$. We can visualize $S(\hat{\boldsymbol{p}})$ by constructing and displaying isosurfaces for various isovalues. These surfaces allows us to display important information related to dependencies between the chosen parameters. Interactive controls allow a user to quickly step through all values of $S(\hat{\boldsymbol{p}})$, resulting in visual cues about the stability of the model.

Figure 2 displays this technique for a simple model of three parameters and two variables. A machine generated description of this model is given in appendix A. [1] In this particular case, the computational domain is a regular 64x64x64 grid. Each point in the domain corresponds to a parameter vector, $\boldsymbol{p}$. Each iso-surface represents one value of the function, $S(\boldsymbol{p})$. The left figure depicts the isosurface when $S(\boldsymbol{p}) = 0.4$. The figure to the right depicts the isosurface when $S(\boldsymbol{p}) = 0.7$.



[1] The calculation of $S(\mathbf{p})$ for every point in the 64x64x64 space took almost eight days on a Silicon Graphics 4D-RPC workstation. Given sufficient memory and graphics performance, an isosurface can be constructed and displayed from this data set in near real-time.

FIGURE 2. Isosurfaces of two values of $S(\boldsymbol{p})$.

Four dimensional parameter spaces are visualized with attribute mapping techniques. This is accomplished with a two step algorithm :

1. For every point in the computational domain calculate $S(\hat{\boldsymbol{p}}_1)$ in which $\hat{\boldsymbol{p}}_1 = (p_1, p_2, p_3, p_4)^{\mathrm{T}}$ and p_4 is a constant value. Construct an isosurface from this data. This is done in the same way as sketched out above.

2. For every vertex on the isosurface calculate $S(\hat{\boldsymbol{p}}_2)$ in which $\hat{\boldsymbol{p}}_2 = (x, y, z, p_4)^{\mathrm{T}}$ and (x, y, z) is a vertex point of the isosurface. Define a transfer function to map these values onto a color as :

$$S(\hat{\boldsymbol{p}}_2) \mapsto \{Red(S(\hat{\boldsymbol{p}}_2)), Green(S(\hat{\boldsymbol{p}}_2)), Blue(S(\hat{\boldsymbol{p}}_2))\}, \tag{5}$$

in which :

$$Red : \mathbb{R} \mapsto [0, 1], \tag{6}$$

$$Green : \mathbb{R} \mapsto [0, 1], \tag{7}$$

$$Blue : \mathbb{R} \mapsto [0, 1]. \tag{8}$$

Linear interpolation of the vertex colors over the polygon results in smooth color transitions over the surface. In this way we see the variation of the fourth parameter at all locations where $S(\hat{\boldsymbol{p}}_1)$ is equal to a particular threshold value.

Iconic mapping techniques can be used to map higher dimensional spaces onto isosurfaces.

3.2. Interactive Visualization

For the representation techniques given above, we assumed to have all the calculated data at hand when applying the visualization algorithms to the data set. Unfortunately, for practical problems, this is not feasible due to :

- the sheer amount of data and the CPU time to calculate this data.
- "the curse of dimensionality" [7]; i.e. points in high-dimensional space tend to be very nearly equidistant, thus confounding distance-based measures of structure detection.

To overcome these drawbacks, an octree is used to represent the computational domain. An octree is a multi-resolution representation of the original computational domain and can be constructed by adaptively subdividing complete domain into small domains. In general, octrees can significantly improve performance of algorithms that repeatedly search large computational domains,

particularly if the interesting regions are sparse and "lumpy". Isosurfaces fit this description. [2]

Three operations are required for displaying isosurfaces with octrees :

1. Build a small part of the octree. An initial octree of relatively few nodes is built in which each node represents some part of the computational domain. The tree is built top down and the root represents the complete computational domain. This region is divided into eight equally sized sub-regions - by dividing each dimension by two - to produce the root's children. This procedure is applied a few times to generate the initial tree.

   Building an initial octree is necessary each time three new parameters are selected.

2. Calculate $S(\boldsymbol{p})$ for each of the eight corner points, $\hat{\boldsymbol{p}}$, in each node of the octree.

   In addition, every node contains a variable indicating the error associated with that node. Although this error metric can be freely chosen, the current implementation uses the following relation :

$$e_k^l = \max\{|S(\boldsymbol{p}_i) - S(\boldsymbol{p}_j)| \parallel i,j = 1,2,\ldots 8\} \tag{9}$$

   i.e. the error of node k at level l of the octree is the maximum of the differences of minimum function at each point of the node.

   Other error metrics may be chosen without effecting the algorithm.

3. An octree walker.

   An adaptive algorithm will traverse a path among a subset of nodes in the octree. The deepest nodes in the full octree correspond to the cubical cells in the Marching Cubes.

The basic idea is that the user will, after supplying an iso-threshold value and an error value, interactively steer the tree walker with an input device (the mouse with the current hardware) through the octree. On each level of the octree an isosurface will be constructed by calculating $S(\boldsymbol{p})$ on the fly for each of the eight points in the region. At each level in the octree, the user supplied error is compared to the calculated error at given node. If the calculated error is less than the user supplied error, the traversal is terminated, otherwise it proceeds downward. If a leaf node is reached and the calculated error is larger than the user supplied error, the node will be divided into eight sub-regions and the traversal will continue.

The result is that a user, by using the mouse to navigate through the computational domain to construct an isosurface, can interactively gain insight in

[2] Pathological cases exist in which isosurfaces fill nearly the entire computational domain so that performance will actually deteriorate when an octree is used.

the behavior of the model. The user may adaptively examine regions of the computational domain at any level of detail.

Figure 3 illustrates this process. The figure shows how a cursor is used to navigate through the computational domain. The cursor, depicted as a small box in the center, is used to steer the numerical engine in the region of interest. An isosurface will be constructed in this region. The figure also depicts other parts of the isosurface. Note that at all times a user can adjust the parameters of the navigation process.

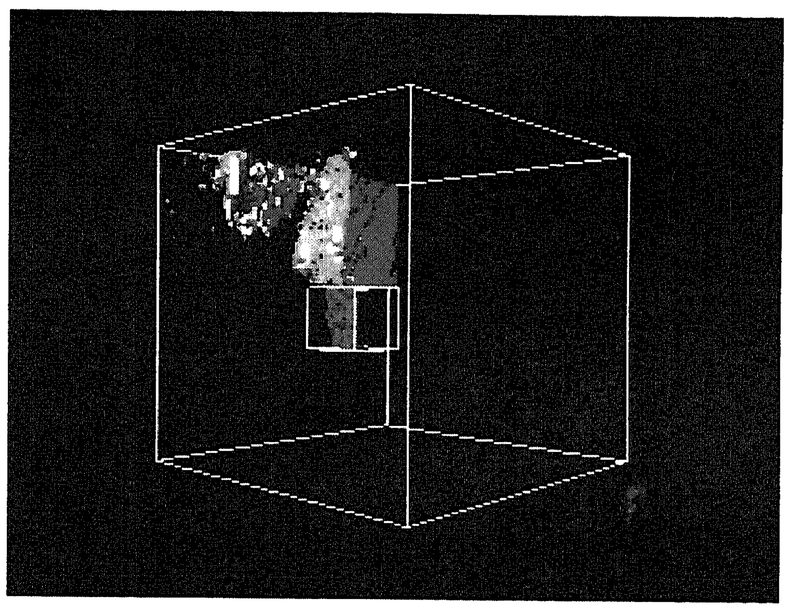


FIGURE 3. Interactive traversal of the octree.

Recently, many visualization researchers have used multi-resolution approximation schemes, [3], [6], to selectively prune the computational space according to a user defined error. Similar techniques are used in various "progressive refinement" computer graphics algorithms.

3.3. Discussion

Exploration of any non-linear system is a heuristic operation. Slight changes in the parameter space often yield unpredictable results. As the exploration proceeds towards to understanding the behavior of a particular problem, many sub-domains of the model's local behavior must also be investigated. In many cases, determining which parameter ranges are interesting is very much a black art. It is quite common to investigate a stable sub-domain and suddenly make a transition to a saddle point or cusp of the surface without being aware of the change in behavior. It may, or may not show up in the simulation trajectories, and can simply appear as a "did not converge" message from the solver.

More valuable would be to provide the chemist with a set of tools to achieve some sort of *qualitative* understanding of the reaction mechanism. We combine numerical simulations with a variety of interactive visualization techniques in order to characterize reaction mechanisms in terms that are meaningful to the

working chemist. The key idea here is that the chemist can use the qualitative characterization of a given mechanism's behavior to determine which steps within the mechanism have an important effect, and which are superfluous and may therefore be dropped from the numerical calculation. With the appropriate visual aids, this information can be fed directly into the solver.

4. Conclusion

We have presented a number of illustrative examples in which a user directly interacts with a complex numerical computation. The primary interface for this interaction are images and the subsequent input based on the user's interpretation of the depicted image. This type of interaction introduces a new category of computational tools for scientists and engineers in which modeling and simulation are tightly integrated with interactive computer graphics.

Acknowledgement

All work on the numerical methods used in the solver was done by Piet Hemker and Jan Kok of the Department of Numerical Mathematics at CWI. Their help and insight on interactive computing is much acknowledged.

Valentein van Dijk has helped in surveying various higher-dimensional representation techniques.

References

1. P.W. Hemker (1972). Numerical methods for differential equations in system simulation and in parameter estimation. In Proceedings: *Analysis and simulation*, 59–80. North-Holland.
2. P.W. Hemker (1972). Parameter estimation in non-linear differential equations. *Technical Report MR 134/72*, Mathematisch Centrum.
3. D. Lear and P. Hanrahan (1991). Hierarchical splatting : A progressive refinement algorithm for volume rendering. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):285–289.
4. W. Lorenson and H. Cline (1987). Marching cubes: A high-resolution 3d surface construction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):163–169.
5. G. Nielson (1991). Interactive visualization. N. Thalmann and D. Thalmann, editors, *New Trends in Modeling and Visualization*, 135–151. Springer Verlag.
6. P. Ning and L. Hesselink (1991). Adaptive isosurface generation in a distortion-rate framework. *SPIE Conference ('91 Proceedings)*, 1459:11–21.
7. E.R. Tufte (1983). *The Visual Display of Quantitative Information.* Graphics Press.
8. C. Upson (1991). Volumetric Visualization Techniques. D. Rogers and R. Earnshaw, editors, *State of the Art in Computer Graphics*, 313–350. Springer Verlag.

Appendix A

This appendix provides a machine generated description of the model used in Section 3 of this article. [3] The model is due to Barnes, [2].

The equations are

$$DF_1 = p_1 Y_1 - p_2 Y_1 Y_2$$

$$DF_2 = p_2 Y_1 Y_2 - p_3 Y_2$$

The Jacobian df/dy is

$$\begin{bmatrix} p_1 - p_2 Y_2 & -p_2 Y_1 \\ p_2 Y_2 & p_2 Y_1 - p_3 \end{bmatrix}$$

The Jacobian df/dp is

$$\begin{bmatrix} Y_1 & -Y_1 Y_2 & 0 \\ 0 & Y_1 Y_2 & -Y_2 \end{bmatrix}$$

The maximal values for y are

$$YMAX_1 = 1.0$$

$$YMAX_2 = 1.0$$

The starting values for y are

$$Y_1 = 1.0$$

$$Y_2 = 0.3$$

The starting values for dy/dp are

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Here only the possible non-zero derivatives of the initial conditions with respect to the parameters are mentioned.

---

[3] This is part of a translation scheme which generates LaTeX and `f77` representations from a high level Maple description. The translation scheme is due to Dr. P. Hemker and Mr. A. Peide.